Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Hochschule für Technik und Wirtschaft

Fachbereich Wirtschaftswissenschaften II

Studiengang Angewandte Informatik

Masterarbeit

Konzeption und Prototyp einer intelligenten Arbeitssuche nach persönlichen Fähigkeiten

---

vorgelegt von Sergej Meister
Matrikelnummer: s0521159

Erstgutachter: Prof. Dr. Christian Herta

Zweitgutachter: Prof. Dr.-Ing. habil. Dierk Langbein

Berlin, 3. Februar 2016

Vorwort

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Was versteht man unter dem Begriff *Information*? Was sind *Daten*? Die beiden Fragen sind sehr beliebt in der Informatikstudium, besonders im ersten Semester. Doch die Informationen und Daten spielen nicht nur in der Informatik sondern in allen Gebieten eine wichtige Rolle. Täglich treffen die Menschen eine Entscheidung auf der Basis Ihres Wissensdaten, aus denen Informationen extrahiert werden. Dadurch dass Informationen aus den Daten gelesen werden, hängt die Entscheidungsfaktor wesentlich von dem *„Qualität*" den bereitgestellten Daten ab.

In der heutigen Informationsgesellschaft durch das schnelle Wachstum von digital verfügbaren Daten ist es für die Menschen schwieriger geworden, die entscheidungsrelevante Informationen aus den Datenmengen zu filtern. Deswegen bietet das Internet viele verschiedene Suchportals für unterschiedliche Interessengruppe, die eine gezielte Suche nach bestimmten Datenmengen erlauben. Einer davon sind Job Portals.

Das Job Portal unterstützt ihrer Anwender rund um das Thema *„Arbeitsuche*" und *„Bewerbungsprozess*". Ein Arbeitssuchender legt Suchkriterien, wie Berufsbeizeichnung und Region, fest und bekommt in weniger Sekunde eine Liste von Arbeitsangeboten, die den Suchkriterien entsprechen. Es ist offensichtlich, dass die Arbeitssuche dank Job Portals viel effizienter und einfacher geworden ist. Trotzdem muss der Arbeitssuchender immer noch viele Informationen selbst raus finden und interpretieren. Zum Beispiel für die Entscheidung, ob die Stelle den gewünschten Anforderungen entspricht, sind die Daten wichtig, wie Berufsname, Firmenname und Qualifikation. Für das Bewerbungsschreiben werden noch weitere Daten benötigt, wie Kontaktperson, Mailadresse, Firmenhomepage und Firmenadresse. Diese Informationen muss der Arbeitssuchender ständig selbst rausfinden. Genau mit dieser Problematik beschäftigt sich das Programm *„Intellijob*", das diese Daten automatisch aus dem Arbeitsangebot extrahiert.

Das Programm *„Intellijob*" wurde im Rahmen der Veranstaltung *„Informationssystemen*" unter Betreuung von Professor Christian Herta entwickelt und kann bereits Berufsbezeichnung, Mailadresse, Firmenhomepage, Kontaktperson und Firmenadresse automatisch auslesen und in Datenbank speichern. Außerdem verfügt das Programm über eine Weboberfläche, die die gespeicherte Daten tabellarisch darstellt.

Der weitere Nachteil von Job-Portals ist die Implementierung von der Suche, die die in der Profile gespeicherte personalisierte Daten nicht berücksichtigt. Denn die typische Suchkriterien eines Job Portals sind einfache Stichworte wie Softwareentwickler, Java, C♯ u.s.w. Daraus wird es ersichtlich, dass die Suche nach einem konkreten Beruf erfolgt. Dieses Suchverfahren eignet sich gut für Quereinsteiger und für die Menschen, die auf der Suche nach neuer Berufsorientierung sind, aber nicht für die Menschen, die eine Arbeit entsprechend ihren Fähigkeiten und Qualifikationen suchen. Wäre es nicht besser, wenn die Suche nach persönlichen Fähigkeiten und Qualifikationen erfolgte, die von Benutzer selbst bewertet werden? Nehmen wir an, der Mensch bewertet seine Java-Kenntnisse mit 2 und C♯ mit 5 Sternen. Dann ist es doch logisch, dass alle C♯ Arbeitsangebote in der Ergebnisliste zuerst angezeigt werden müssen. Vielleicht ist es auf dem ersten Blick nicht ganz ersichtlich, woran die beide Verfahren sich unterscheiden. Die Suche nach einem Beruf als Softwareentwickler kann aber die Java- Arbeitsangebote zuerst anzeigen, was aus der Sicht der Arbeitssuchender mit den besseren C♯ Kenntnissen nicht ganz zutreffend ist. Der weitere Unterschied liegt im Entscheidungsfaktor. Bei dem ersten Verfahren entscheidet der Mensch, welche Arbeit er haben will. Bei dem zweiten Verfahren entscheidet das Programm, welche Arbeit für den Menschen am besten geeignet ist.

Die Untersuchung von mehreren Job-Portals wie *„Xing"*, *„StepStone"*, *„JobScout24"*, *„Monster"* hat bestätigt, dass es tatsächlich kein deutsches Job-Portal gibt, das die Suche nach persönlichen Fähigkeiten und Qualifikationen unterstützt.

1.2 Ziele der Arbeit

Im Rahmen dieser Masterarbeit wird ein Suchverfahren implementiert, dass die Suche nach persönlichen Fähigkeiten und Qualifikationen unterstützt. Die Suche muss mit *Elasticsearch*[1] realisiert und in der Anwendung *IntelliJob* integriert werden. Um die Suchqualität bewerten zu können, müssen mehrere unterschiedliche Suchqueries gebildet und genau untersucht werden. Am Ende der Arbeit wird geprüft, ob das neue Suchverfahren gegenüber der traditionelle Suche nach Stichworte eine bessere Informationsmenge liefert. Das Suchverfahren wird als effektiv bewertet, wenn die erste Arbeitsangebote in der Ergebnisliste den persönlichen Fähigkeiten und Qualifikationen besser entsprechen werden.

Das weitere Thema dieser Arbeit wird die Datenbeschaffung sein. Um nach persönliche Fähigkeiten und Qualifikationen suchen zu können, müssen diese Daten zuerst erfasst werden. Dafür wird das Projekt *„Intellijob"* um eine neue Web-Form erweitert, die die Bewertung von persönlichen Fähigkeiten und Qualifikationen ermöglicht. Die Daten, die dem Arbeitssuchender bereitgestellt werden, müssen sinnvoll gruppiert werden, so dass die Kriterien wie Java und C♯ unter den Sammelbegriffen Programmieren, Softwareentwickeln, gefunden werden könnten. Der Arbeitssuchender muss in der Lage sein, seine eigene Daten speichern, editieren

[1] **www.elastic.co**.

und löschen zu können. Außerdem darf der Benutzer selbst entscheiden, welches Suchverfahren eingesetzt werden soll.

1.3 Aufbau der Arbeit

Der schriftliche Teil der Arbeit beschreibt das komplette Entwicklungsprozess. Im nächsten Kapitel *„Grundlagen"* werden die Technologie erläutert, die für die Entwicklung neues Suchverfahrens relevant sind. Dabei wird das Projekt *IntelliJob* und die eingesetzte Technologie kurz vorgestellt. Das Kapitel *„Systementwurf"* befasst sich mit Softwarearchitektur und beschreibt die Logikverteilung und die Datenkommunikation innerhalb der Anwendung. Das vierte Kapitel *„Entwicklung und Implementierung"* stellt das Prototyp des neuen Suchverfahrens vor. Dabei wird sowohl das Datenmodel als auch die wichtigste Funktionen detailliert beschrieben. Außerdem wird in diesem Kapitel auch die API-Schnittstelle dokumentiert. Im Unterkapitel *„Benutzeroberfläche"* wird die Web-Form zur Erfassung von persönlichen Fähigkeiten und Qualifikationen als Bild dargestellt. Das Kapitel *„Evaluierung"* analysiert mehrere unterschiedliche Suchanfragen und bewertet sie. Problemen und Schwierigkeiten, die während der Implementierung aufgetreten sind, sowie die Art und Weise, wie diese Probleme gelöst oder eventuell nicht gelöst wurden, werden in dem 6. Kapitel erfasst. Zum Schluss wird die Arbeit zusammengefasst und ein Resümee gezogen. Ein Ausblick zeigt auf, welche Erweiterungen für die Anwendung in Zukunft noch sinnvoll sein könnten.

2. Grundlagen

In diesem Kapitel wird auf die allgemeinen und informatischen Grundlagen eingegangen, welche zum Verständnis dieser Masterarbeit relevant sind. Die Anwendung *IntelliJob* persistiert bereits die Daten in eine dokumentenorientierte Datenbank *MongoDB*[1] und es wird noch eine weitere dokumentenorientierte Technologie *Elasticsearch* eingesetzt. Deswegen ist es wichtig mindestens die Grundlagen sowie die Vor- und Nachteile von dokumentenorientierten Datenbanken im Blick zu behalten. Die in der Arbeit verwendete *Elasticsearch-Feature's* werden in einem extra Kapitel ausführlich beschrieben. Da die Suchfunktionalität in der bereits vorhandene Anwendung integriert werden muss, ist es für die Bewertung der Arbeit notwendig den IST-Zustand zu dokumentieren.

2.1 Dokumentenorientierte Datenbanken

Bei dokumentenorientierten Datenbanken werden die Daten in Form von Dokumenten gespeichert. Die typische Dokument-Formate sind XML, YAML und JSON. Die einzelnen Datenfelder werden als Key-Value Stores zusammengefasst. Jedes Dokument ist vollkommen frei bezüglich seine Struktur und das ist ein wesentlicher Unterschied zu relationalen Datenbanken, wo die Datenstruktur vordefiniert ist [Dorschel 2015]. Die Abbildung 1. zeigt eine typische normalisierte Datenstruktur in den relationalen Datenbanken. Einige Datensätze in der Tabelle *skills* beinhalten kein *"description"*. Die Spalte muss in dem Fall einfach leer bleiben. Es besteht keine Möglichkeit nur für bestimmte Datensätze auf die Spalte *"description"* zu verzichten.

category_id	name
1	Languages
2	Knowledges
3	Personal strengths

skill_id	category_id	name	description
1	1	German	
2	1	English	
3	2	Java	programming language java
4	2	PHP	programming language php

Abbildung 2.1: Relationale Tabellen *skill-categories* und *skills*

In dokumentenorientierten Datenbanken ist es aber durchaus möglich, dass einige Dokumente innerhalb einer *Collection* kein *description* Feld haben(siehe Abbildung 3).

[1] **www.mongodb.org**.

```
{
  "_id" : ObjectId("569ab61a44ae6344028b6cb5"),
  "name" : "Languages"
},
{
  "_id" : ObjectId("569ab65644ae6344028b6cb7"),
  "name" : "Knowledges"
},
{
  "_id" : ObjectId("569ab66d44ae6344028b6cb9"),
  "name" : "Personal strengths"
}
```

Abbildung 2.2: Mongo Collection *skill-categories*

```
{
  "_id" : ObjectId("569ab85144ae6344028b6cbf"),
  "category" : {
     "_id" : ObjectId("569ab61a44ae6344028b6cb5"),
     "name" : "Languages"
  },
  "skills" : [
     {
        "_id" : ObjectId("569ab8dd44ae6344028b6cc0"),
        "name" : "deutsch"
     },
     {
        "_id" : ObjectId("569ab8dd44ae6344028b6cc1"),
        "name" : "english"
     }
  ]
},
{
  "_id" : ObjectId("569ab9d644ae6344028b6cc9"),
  "category" : {
     "_id" : ObjectId("569ab65644ae6344028b6cb7"),
     "name" : "Knowledges"
  },
  "skills" : [
     {
        "_id" : ObjectId("569ab9d644ae6344028b6cca"),
        "name" : "java",
        "description" : "programming language java"
     }
  ]
}
```

Abbildung 2.3: Mongo Collection *skills*

Wenn man die Abbildungen 2 und 3 genau anschaut, fehlt es sofort auf, dass die *Category-Daten* nicht redundant sind. In den relationalen Datenbanken werden die Datenattribute durch die Anwendung von verschiedenen Normalisierungsregeln solange zerlegt, bis keine vermeidbaren Redundanzen mehr vorhanden sind. Die dokumentenorientierte Datenbanken verfolgen den Prinzip der *Aggregation*. Der Begriff kommt eigentlich aus dem Domain-Driven Design und bedeutet, dass jedes Dokument eine eigenständige Einheit ist, das von keinen anderen beziehungsweise von möglichst wenigen Dokumenten abhängig ist. Dadurch können die Daten immer vollständig automar gespeichert oder gelesen werden.

Diese hohe Datenflexibilität ist aber nicht umsonst. Zum Ersten ist es oft aufwendig die Daten zu ändern, die bereits in anderen Dokumenten gespeichert sind. Ein Beispiel wäre dafür das Dokument mit dem Name *„Languages"* in der *Collection „skill-categories"* (Abbildung 2.2) mit einem neuen Attribute zu erweitern oder komplett zu löschen. In dem Fall muss diese *„Category"* auch in allen anderen Dokumenten gefunden und entsprechend geändert werden(Abbildung 2.3). Zum Zweiten können dokumentenorientierte Datenbanken aufgrund ihrer Schemafreiheit keine einfache Datenvalidierung der Attributen und Datentypen durchführen. Das muss stattdessen im Programmcode behandelt werden, was zu höheren Softwarekosten führt und fehleranfällig ist [Jan 2013].

Wenn man das ganze kurz wörtlich zusammenfasst: Schemafreiheit, keine Datenvalidierung, mehr Programmcode, Fehleranfällig und aufwändige Datenänderung auf der Datenbankebene, fragt man sich, wieso werden aber die dokumentenorientierte Datenbanken, die seit 2008 unter dem Titel *„NoSQL"*[2] bekannt sind, immer beliebter? Eine einfache Antwort lautet, dass die NoSQL-Datenbanken oft schneller als relationale Datenbanken sind. Statt mehrere Tabellen mit JOIN-Operation abfragen zu müssen, führt ein NoSQL-Datenbank nur eine einzige Abfrage aus, um ein kompletter Datensatz zu erhalten. Die Datenkonsistenz beim Lesen und Schreiben in mehrere Tabellen wird in klassischen relationalen Datenbanken durch Transaktion-Mechanismen realisiert. Jede Transaktion ist isoliert von einander und wird ganz oder gar nicht ausgeführt. Wenn eine Transaktion in eine Tabelle schreibt, kann keine andere Transaktion auf die Tabelle zugreifen, bis der Schreibvorgang komplett abgeschlossen oder unterbrochen wird. Das führt dazu, dass die Lese- und Schreiboperation von parallel ausgeführten Transaktionen einander blockieren und warten müssen, bis alle nötige Ressource freigegeben werden. Dadurch dass NoSQL Datenbanken keine Abfrage blockieren, werden die auch schneller abgearbeitet. Es besteht zwar die Möglichkeit, dass inkonsistente Daten gelesen werden, aber aufgrund, dass die Daten oft nur aus einer einzigen *Collection* gelesen werden, ist die Fehlerwahrscheinlichkeit enorm klein im Vergleich zu viel steigender Performance[Dorschel 2015]. Die einzige Schutzmaßname, die von meisten NoSQL-Datenbanken unterstützt werden, ist die Datenversionierung. Bei jedem Speicherzugriff wird die Version des Dokumentes geprüft, ist die Version identisch dem bereits gespeicherten Dokument, dann wird ein *Exception* geworfen. Denn es ist ein Zeichen dafür, dass Dokument bereits von einem anderen *Request*

[2] **https://de.wikipedia.org/wiki/NoSQL**.

geändert wurde. Außerdem wird auch die Datenversionierung für die Synchronisierung zwischen mehrere Rechnerknoten innerhalb eines *Clusters* verwendet. Das *Sharding Concepts* in dokumentenorientierten Datenbanken wird im Kapitel über *Elasticsearch* detailliert beschrieben.

Fazit: Die dokumentenorientierte Datenbanken eignen sich besonders gut für die Anwendungen, wo hohe Geschwindigkeit bei Datenzugriffe und Datenverarbeitung im Vordergrund steht und die Datenredundanz nicht hoch priorisiert wird. Die meisten Datenbankzugriffe in der Anwendung *IntelliJob* sind Lesezugriffe. Es gibt wenige Möglichkeiten die Daten durch Benutzeraktion zu ändern. Die Änderungen beeinflussen nur eine einzige *Collection* und oft sogar nur ein einziges Dokument. Deswegen ist die Entscheidung über den Einsatz von dokumentenorientierten Datenbanken richtig.

2.2 IntelliJob

Die Anwendung *„IntelliJob"* wurde bereits mehrmals erwähnt und muss endlich vorgestellt werden. Es gab mal eine Idee die bewerbungsrelevante Daten aus Arbeitsangeboten automatisiert zu extrahieren. Diese Idee wurde mit dem Programm *„IntelliJob"* realisiert. Da die Anwendung im Rahmen der Veranstaltung *„Informationssystemen"* eigenständig entwickelt und dokumentiert wurde, werden einige Inhalte aus der Arbeit in diesem Kapitel zitiert.

Bevor die Daten aus dem Arbeitsangebot gelesen werden könnten, muss dieses Arbeitsangebot erstmals ermittelt werden. Dafür werden mehrere Job-Agenten auf Job-Portals *Stepstone*[3] und *Monster*[4] angelegt, die die Arbeitsangebote per Email täglich senden. Das Email beinhaltet mehrere HTML-Links mit unterschiedlichen Arbeitsangeboten. Die Anwendung *„IntelliJob"* holt diese Emails aus dem Postfach ab, ruft die Links auf und speichert das geladene Inhalt in Datenbank ab.

Im nächsten Schritt werden die bewerbungsrelevante Daten, *Berufsbezeichnung*, *Firmenadresse*, *Firmenhomepage*, *Mailadresse* und *Kontaktperson*, aus dem Arbeitsangebot extrahiert.

Berufsbezeichnung wird einfach aus dem HTML-Link ausgelesen, das als *Value* immer eine Berufsbezeichnung enthält.

Mailadresse und **Firmenhomepage** werden mit Hilfe von Regulären Ausdrücken extrahiert.

Mail-Pattern: *[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+*

WWW-Pattern: *www.[\w\d.:#@%/;$()~_?+-=&]**

HTTP-Pattern: *http.[\w\d.:#@%/;$()~_?+-=&]**

[3] **www.stepstone.de**.
[4] **www.monster.de**.

HTTPS-Pattern: *https.[\w\d.:#@%/;$()~_?+-=&]**

Leider können Reguläre Ausdrücke nur Daten mit einer eindeutigen Struktur ermitteln. **Kontaktperson** und **Firmenadresse** können aber sehr unterschiedlich geschrieben werden:

- Alle der Kosmonauten 26a D-32451 Bad Kreuznach
- Straße des 17. Juni 17a 13456 Berlin (Lieblings Beispiel)
- Herr Richard-Alexander Wagner
- Johann Wolfgang von Goethe
- Alexander Sergejewitsch Puschkin

Aus den oben aufgelisteten Beispielen wird es ersichtlich, dass es kaum möglich ist, allein mit Regulären Ausdrücke, diese Daten zu extrahieren. Deswegen werden diese Daten in *„IntelliJob"* mit Hilfe von *„Apache OpenNLP Framework"*[5] gefunden und zusammen mit allen anderen bewerbungsrelevanten Daten in Datenbank gespeichert.

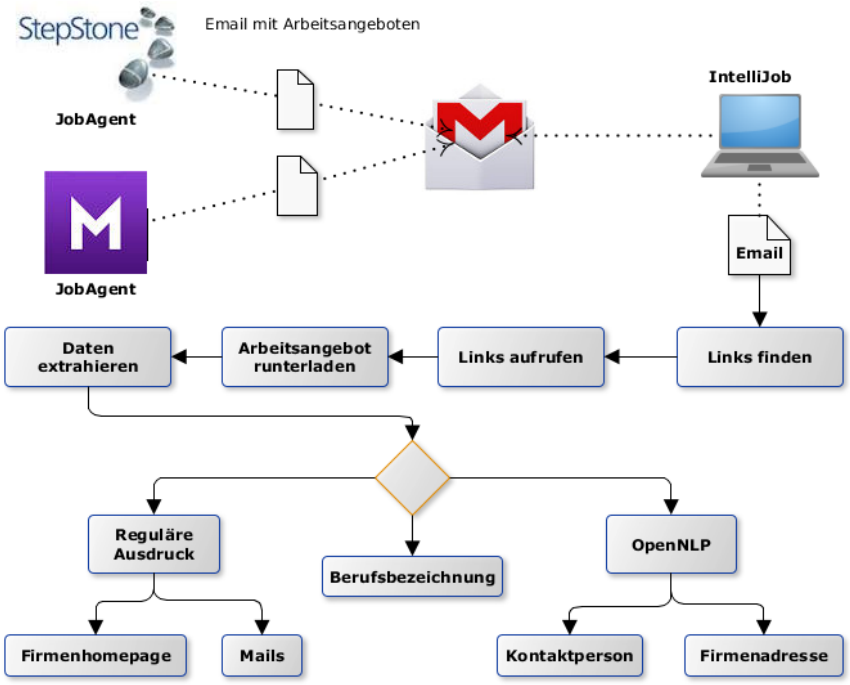

Abbildung 2.4: Prozessablauf in IntelliJob

[5] **https://opennlp.apache.org**.

2.2.1 Technologie Stack

Apache OpenNLP ist ein Open Source Produkt und wird als Maven-Dependency in der Version 1.5.3 in das Projekt *„IntelliJob"* eingebunden. Das Framework setzt die Technologie aus dem Gebiet *Natural Language Processing* ein, um bestimmte Bestandteile eines Textes zu erkennen und zu klassifizieren. Die Muster-Erkennung basiert auf künstlich erzeugten Trainingsdaten, die durch ein Lernprozess in einem Model gespeichert werden. Das Framework wird ständig weiter entwickelt und verfügt über eine gute Dokumentation.[6]

Die Anwendungslogik von *„IntelliJob"* wurde in der Programmiersprache Java geschrieben und verwendet sehr viele Komponenten aus dem Spring Framework[7]. Spring Framework ist ebenfalls ein Open Source Produkt, welches performante und moderne Java-Enterprise Einsätze realisiert. Mit dem Einsatz von Spring *Dependency Injection* und *Aspekt orientierte Programmierung* wird das Programmcode redundant verteilt und kann auch leicht wiederverwendet werden. Alle erzeugte Objekte werden anhand der *Spring Data*[8] in eine dokumentenorientierte Datenbank MongoDB persistiert.

Die weitere Besonderheit von *„IntelliJob"* ist, dass es eine Standalone Web-Anwendung ist, die lokal auf dem Client einen embedded Web Container *Tomcat*[9] startet. Der große Vorteil von diesem Einsatz liegt darin, dass der Webserver ein Teil der Anwendung ist, und die Anwendung nicht mehr in den Webserver deployed werden muss. Das Programm wird als JAR-Datei ausgeliefert und kann direkt in der Java Virtual Machine ausgeführt werden. Realisiert wird das ganze mit *Spring Boot*[10] Technologie, die dank Autokonfiguration sowohl alle nötige Resource zusammenpackt, als auch alle Rest-Services zur Kommunikation mit Frontend initialisiert.

Das Frontend wird mit *AngualrJS*[11] umgesetzt und verfügt bereits über 6 verschiedene Views. Das Framework unterstützt die Entwickler bei der Implementierung von Single Page Anwendungen.

Home	Startseite um Mail-Account auszuwählen und Zugangsdaten zu übergeben.
Emails	zeigt alle gefundene Emails von Job Portals
Job Links	zeigt alle in Mails gefundene Links zu Arbeitsangeboten
Jobs	zeigt alle runter geladene Arbeitsangebote
Jobs Details	zeigt alle extrahierte Daten eines Arbeitsangebotes
Audit	dient zur Datenauswertung und zeigt sowohl die aktuellste als auch die alte Ergebnisse des Datenextraktions.

Während der Masterarbeit wird *AngularJS* von der Version 1.2.16 zu 1.4.7 aktualisiert.

[6] **https://opennlp.apache.org/documentation/1.6.0/manual/opennlp.html**.
[7] **www.spring.io**.
[8] **http://projects.spring.io/spring-data**.
[9] **http://tomcat.apache.org**.
[10] **http://projects.spring.io/spring-boot**.
[11] **https://angularjs.org**.

2.2.2 Software Architektur

„IntelliJob" ist quelloffen und kann auf Versionsverwaltungsplattform, GitHub[12], heruntergeladen werden. Die Anwendungslogik basiert auf Schichtenarchitektur, die sequenziell von oben nach unten abgearbeitet wird.

Präsentationsschicht repräsentiert die Daten in HTML-Form und regelt die Interaktionen zwischen Benutzer und Software. Alle Benutzeranfragen werden asynchron an Serviceschicht weitergeleitet. Die Daten zwischen beiden Schichten werden in JSON-Format transportiert.

Serviceschicht nimmt die Client-Anfragen entgegen, validiert Benutzerdaten und wandelt sie in Domain-Objekte um. Danach wird die Abfrage der Controlschicht übergeben. Letztendlich wird ein *Response* generiert und zurückgesendet.

Controlschicht vereint alle Methoden aus Datenzugriffsschicht und führt alle nötige Operationen aus, um die Benutzeranfrage zu bearbeiten.

Datenzugriffsschicht kapselt die Zugriffe auf persistente Daten. Die Datenaustausch erfolgt über Domain-Objekte.

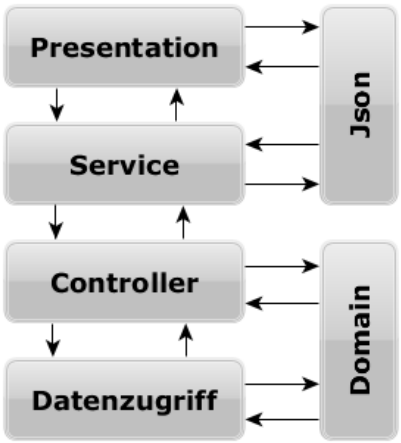

Abbildung 2.5: Schichtenarchitektur

Dadurch dass die Anwendungslogik in mehrere Schichten verteilt wird, wird die Komplexität der Anwendung wesentlich reduziert, was sowohl für das Verständnis als auch für die Wartung von Software eine große Vorteil ist. Außerdem können die Programmteile dank seiner Abstraktion gut getestet und auch leicht wiederverwendet werden.

Um die Methoden zum Datenextraktion auch in anderen Projekten zu verwenden, werden sie in ein separates Projekt, namens *Civis-Tools*[13] ausgelagert.

[12] **https://github.com/SergejMeister/intellijob**.
[13] **https://github.com/SergejMeister/civis-tools**.

2.2.3 Problembeschreibung

Zum Beginn der Masterarbeit gab es ein kleines Problem in *„IntelliJob*", das früher immer missachtet wurde und während der Masterarbeit unbedingt gelöst werden müsste. Das Problem liegt in doppelt gespeicherten Arbeitsangeboten.

Jedes Mail von Job-Portal beinhaltet Links sowohl mit neuen als auch mit alten Arbeitsangeboten. Deswegen wird es vor dem Schreiben in Datenbank immer geprüft, ob das Link zu diesem Arbeitsangebot schon früher gespeichert wurde. Durch die Beobachtung wurde allerdings erkannt, dass die Links zum gleichen Arbeitsangebot manchmal geändert werden. Außerdem kann ein Arbeitsangebot sowohl von StepStone als auch von Monster Portal kommen. Die Links und Mail sind dabei unterschiedlich und es gibt auch keine logische Möglichkeit die beiden Arbeitsangebote von einander zu unterscheiden.

2.3 Tag Cloud

Abbildung 2.6: Tag Clouds - *www.einfachbewusst.de*

Die Tag Cloud, deutscher Begriff Schlagwortwolke, werden im Web immer beliebter und dienen zur Visualisierung von gewichteten Daten. Je nach Gewichtung werden die Daten in verschiedenen Schriftgröße und Farben dargestellt. Mit Hilfe von Tag Cloud können beispielsweise die meistbesuchte Beitrage eines Weblogs oder die häufigste Worte eines Dokuments angezeigt werden. In der Implementierungsphase dieser Masterarbeit werden die Tag Clouds zur Visualisierung von persönlichen Fähigkeiten in der Suchmaske eingesetzt. Dadurch wird es für den Benutzer ersichtlich, welche Fähigkeiten bei der Suche besonders priorisiert werden.

In der Praxis werden die einzelnen Wörter der Tag Cloud mit einer Unterseite oder Dokument verlinkt, die am relevantesten für diese Schlagwörter sind. So entsteht eine Art alternative zu einem Navigationsmenü. Der Nachteil von diesem Einsatz besteht darin, dass die Gewichtung der Schlagwörter durch Aufruf der Seite oder Dokumentes erfolgt und keine Information darüber enthält, ob der verlinkte Inhalt den Benutzererwartungen entspricht. Der Benutzer klickt auf dem hoch gewichtete Schlagwort an, stellt fest, dass der verlinkte Inhalt nicht den Erwartungen

entspricht und verlässt die Seite. Da die Seite jedoch aufgerufen wurde, wird der Schlagwort der Tag Cloud höher gewichtet[OnPage 2016].

Die Tag Cloud wird meistens nach der folgenden Formel berechnet:

$$S_i = [(f_{max} - f_{min}) * \frac{t_i - t_{min}}{t_{max} - t_{min}} + fmin]$$

- S_i - anzuzeigende Schriftgröße
- f_{max} - maximale Schriftgröße
- f_{min} - minimale Schriftgröße
- t_i - Häufigkeit des betreffenden Schlagwortes
- t_{min} - Häufigkeit, ab der ein Schlagwort angezeigt werden soll
- t_{max} - Häufigkeit des häufigsten Schlagwortes[14]

2.4 Informationsbeschaffung

Im ersten Kapitel wurde schon erwähnt, dass die Informationen aus Daten extrahiert werden. Zur Informationsbeschaffung (eng. Information Retrieval) gehören neben Extraktionsprozess noch weitere Prozesse. Zuerst müssen die Daten erfasst werden und in einer Datenbank oder einem Daten-Management-System in einer bestimmen Syntax abgelegt werden. Dabei legt der Verfasser einen Teil seines Wissen in ein Dokument nieder, das in Text, Bild oder eine andere Form gespeichert wird. Im nächsten Schritt müssen die Daten, die bereits existieren, wiedergefunden und in geeignete Repräsentationsform gebracht werden. Letztendlich sollen nur die Informationen extrahiert werden, die möglichst gut bei einer Problemlösung helfen können. Der Autor Andreas Heinrich in seinem Lehrtext *„Information Retrieval Grundlagen, Modelle und Anwendungen*" bezeichnet dieses Prozess als Wissenstransfers und weist deutlich darauf hin: *„... Information Retrieval nicht nur als den Prozess der Wissensextraktion aus einer Dokumentenkollektion sondern als Prozess des Wissenstransfers zu verstehen*"[Henrich 2008]. Die Abbildung 2.7 stellt das Prozess vereinfacht dar.

Das System, das sich mit der Informationsbeschaffung befasst, wird in der Informatik als IR-System bezeichnet. Die Aufgabe des IR-Systems besteht hauptsächlich darin, die hinter der Anfrage stehende Informationsbedürfnisses zu befriedigen. Oft muss ein IR-System neben der Anfrage noch weitere Datenquellen, wie Feedback oder vorhandene Information über Nachfragender, zu berücksichtigen, um eine bessere Informationsergebnis zu erzielen. Im Bezug auf die Anwendung *„IntelliJob*" kann Informationsergebnis wesentlich verbessert werden, indem der Nachfragender die Suchanfrage selber priorisiert.

[14] **https://de.wikipedia.org/wiki/Schlagwortwolke**.

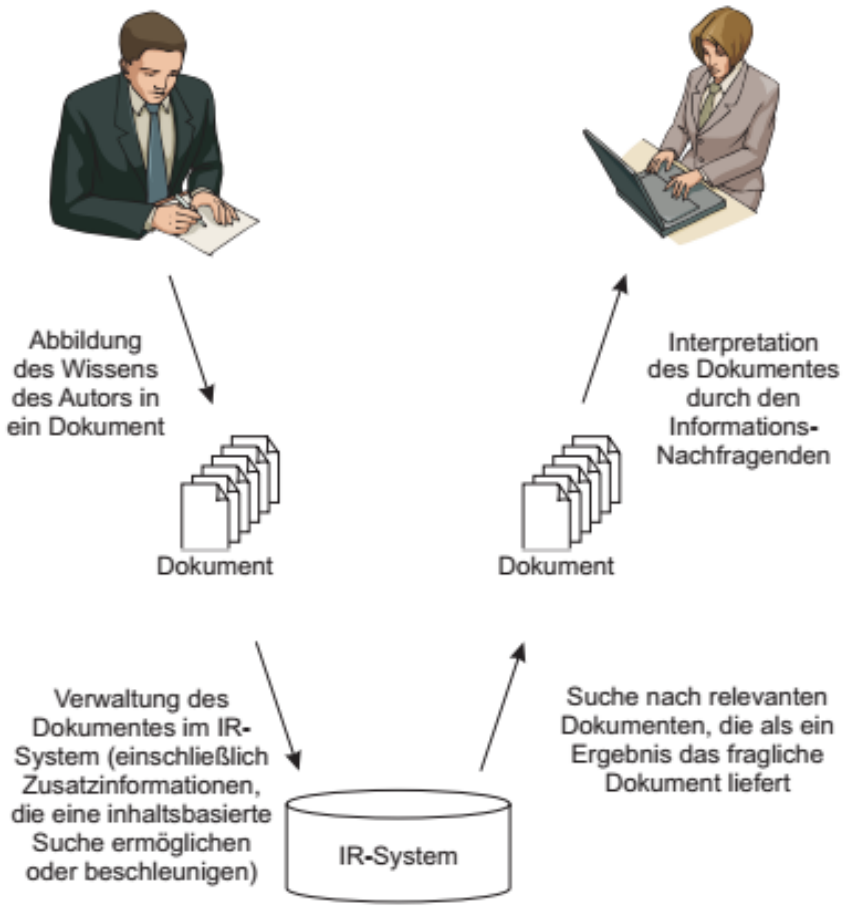

Abbildung 2.7: Prozess des Wissenstransfers vom Autor eines Dokumentes bis zum Informations-Nachfragenden

2.4.1 Qualität in Information Retrieval System

Um ein gutes IR-System von einem schlechtes unterscheiden zu können, muss die Qualität des Ergebnissen gemessen werden. Die Qualität wird ihrerseits durch **Relevanz**, **Recall** und **Precision** bestimmt.

Relevanz kann binär (*relevant/nicht relevant*) oder mit Relevanzeinstufung (*von 1 bis 10*) definiert werden. In der Praxis hat sich allerdings die binäre Relevanz durchgesetzt. Die Entscheidung, ob ein Dokument relevant oder nicht relevant ist, ist sehr subjektiv und erfordert Vorwissen sowohl über das untersuchte Dokument als auch über andere Dokumenten. Deswegen wird es sehr schwierig alle Dokumente einzustufen[Henrich 2008].

Recall beschreibt der Anteil der relevanten Dokumenten in der Ergebnismenge im Verhältnis zum allen relevanten Dokumenten und wird wie folgt berechnet:

$$Recall = \frac{a}{S}$$

- a - Anzahl der relevanten Dokumente im Ergebnis

- S - Gesamtzahl der relevanten Dokumente

Precision zeigt der Anteil nicht relevanten Dokumenten in der Ergebnismenge.

$$Precision = \frac{a}{D}$$

- a - Anzahl der relevanten Dokumente im Ergebnis
- D - Gesamtzahl der Dokumente im Ergebnis

Recall und **Presision** dürfen aber nicht einzeln betrachtet werden. Denn nur der Vergleich von beiden Werten weist deutlich darauf hin, ob ein IR-System besser oder schlechter gegenüber eines anderen Systems ist.

2.4.2 Indexierung von Textdokumenten

Die vorliegende Arbeit beschäftigt sich mit Arbeitsangeboten, die in Textform vorliegen. Diese Textdokumente müssen automatisiert erfasst werden. Dieses Prozess wird in Information Retrieval als *Indexierung* bezeichnet. Bei der Indexierung wird ein Dokument analysiert und alle Wörter außer *Stopwörter* werden extrahiert. Die *Stopwörter* sind besondere Wörter die sehr häufig vorkommen. In der deutschen Sprache sind Artikeln ein typisches Beispiel für die *Stopwörter*. Die extrahierte Worte werden in Grundform (z.B. *schreiben* - *schreib*) gebracht und als Terme abgespeichert. Durch Grundformredaktion wird es verhindert, dass die gleiche aber unterschiedlich geschriebene Worte mehrmals gezählt werden. Denn während der Indizierung wird sowohl das Vorkommen eines Wortes in einem Dokument(term frequency, TF) als auch in allen anderen Dokumenten (inverse document frequency, IDF) gezählt.

$$IDF_i = log\frac{N}{n_i}$$

- i - der untersuchte Term
- N - Anzahl aller Dokumenten
- n_i - Anzahl der Dokumente mit Term

Der Logarithmus sorgt dafür, dass ein oft vorkommender Term nicht extrem hoch gewichtet wird. Das Endergebnis der automatischen Indexierung ist ein Gewicht für jeden Term in jedem Dokument [Mandl 2001]. Nachdem das Dokument erfolgreich indexiert wurde, kann es auch durchgesucht werden.

2.4.3 Boolesche Suche

Die Boolesche Suche oder das Boolesche Retrieval Modell basiert auf die elementare Mengenoperationen, wie Schnittmenge (AND-Verknüpfung), Vereinigungsmenge (OR-Verknüpfung) und Komplementärmenge (NOT-Verknüpfung). Das Modell

geht davon aus, dass das Ergebnis einer Suchanfrage als exakte Menge von relevanten Dokumenten bestimmt werden kann. Dabei werden nur Dokumente ausgegeben, die den in der Suchanfrage formulierten booleschen Ausdruck erfüllen.

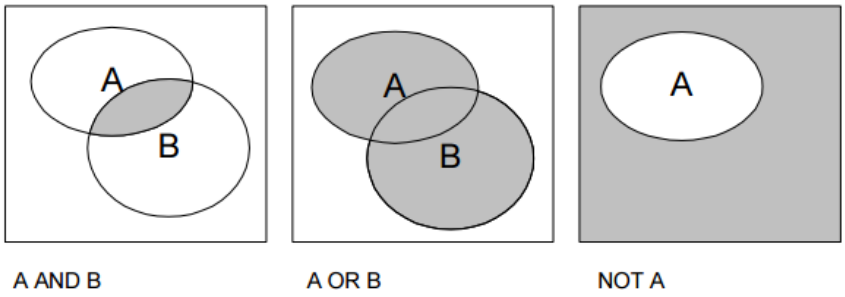

Abbildung 2.8: Die elementaren booleschen Operationen

Der Nachteil des booleschen Modells besteht darin, dass die Gewichtung von Termen nicht berücksichtigt wird und die Ergebnismenge deswegen ungeordnet ausgeliefert wird. Außerdem werden viele Dokumente nicht gefunden, weil die gesuchte Elemente nicht in den exakt gleichen Form in Dokumenten enthaltet sind. Trotz seiner Ungenauigkeit wird das boolesche Model oft eingesetzt vor allem um mehrere Suchanfragen mit einander zu verknüpfen.

2.4.4 Vektorraummodell

Zum ersten Mal wurde das Vektorraummodell von Gerard Salton, Andrew Wong und Chungshu Yang in einem Zeitschriftartikel unter dem Thema *„A Vector Space Model for Automatic Indexing*" im Jahr 1975 publiziert und ausführlich beschrieben[Salton; Wong; Yang 1975]. Die Grundidee dieses Modells besteht darin, dass sowohl die Anfrage als auch alle Dokumente in einem vieldimensionalen Vektorraum dargestellt werden. Dabei bildet jeder einzelne Term eine Achse beziehungsweise eine Dimension in diesem Vektorraum und die Koordinaten werden durch die Gewichtung von Termen repräsentiert.

Die Relevanz eines Dokumentes im Vektorraummodell wird oft durch Berechnung des Cosinus des Winkels zwischen Anfrage und Dokument bestimmt. Das führt dazu, dass nicht mehr nach exakte Übereinstimmungen zwischen Suchanfrage und Dokumenten sondern nach ihre Ähnlichkeit gesucht wird und die Ergebnismenge auch sortiert ausgegeben wird.

Obwohl das Vektorraummodell die wesentliche Nachteile des booleschen Modells aufhebt, kann das boolesche Model unter bestimmte Bedienungen eine bessere Ergebnismenge als Vektorraummodell liefern. Damit die Ähnlichkeitsmaß zwischen Anfrage und Dokument mit Vektorraummodell sinnvoll berechnet werden könnte, muss die Anfrage aus mehrere Begriffe bestehen. Wenn die Suchanfrage aus wenigen Begriffen besteht, kann das boolesche Model, das diese Begriffe einfach mit *AND* verknüpft, oft zu besseren Ergebnissen führen.

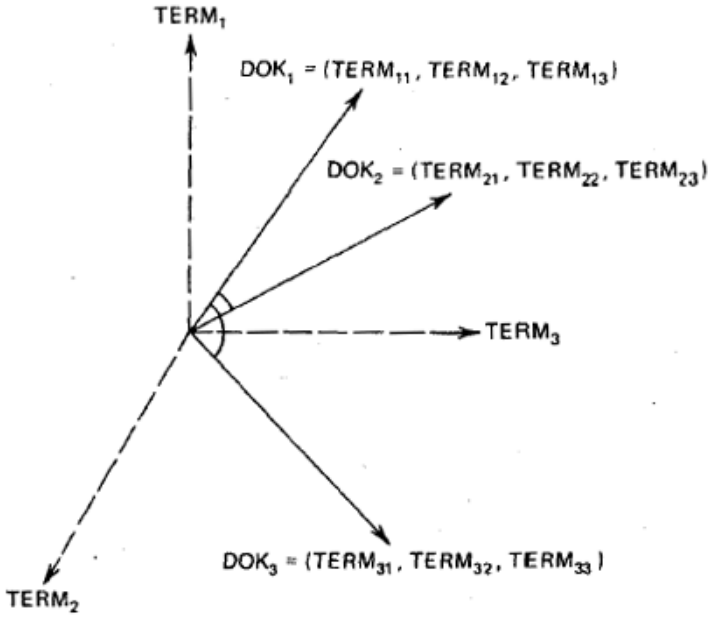

Abbildung 2.9: Verktorraumpräsentation eines Dokumentraums (Salton u. McGill 1987, 129)

2.4.5 Volltextsuche

Bei der Volltextsuche handelt es sich darum, eine benutzerdefinierte strukturierte Suchanfrage in einem unstrukturierten Textdokument zu finden. Natürlich kann man dafür das boolesche oder Vektorraummodell einsetzen, um die erste Ergebnisse zu bekommen. Allerdings bieten die meisten IR-System noch einige Methode für die Optimierung von Suchanfragen, die die Ergebnismenge wesentlich beeinflussen können. Der nachfolgende Abschnitt beschreibt kurz die Optimierungsmethode, die für den praktischen Teil nützlich sein könnten.

Phrasensuche wird für die Suche nach zusammenhängenden Begriffen eingesetzt. Abhängig von Einsatzszenarien kann die Phrasensuche entweder sehr gute oder sehr schlechte Ergebnismenge liefern. Wenn man nach einen Title oder eine exakte Phrase in einem Dokument sucht, dann bekommt man gute Ergebnisse. Allerdings werden viele potenziell relevante Dokumente ausgeschlossen, weil die gesuchte Phrase in diesen Dokumenten nicht in der gleichen Wortfolge vorliegt. Sucht man beispielsweise nach *„relationale Datenbanken*", werden Dokumente mit *„relationale Datenbank*" nicht gefunden.

Fuzzy-Suche sorgt für Fehlertoleranz und erlaubt eine vordefinierte Abweichung zwischen den gesuchten und in Dokument vorhandenen Term. Das bekannteste Algorithmus für die Fuzzy-Suche ist Levenshtein-Distanz[15]. Das Algorithmus berechnet die Anzahl von Änderungen, die nötig sind, um ein Wort in ein anderes umzuwandeln. Je wenig Änderungen notwendig sind, desto ähnlicher sind die Worte.

[15] **https://de.wikipedia.org/wiki/Levenshtein-Distanz**.

Filter werden verwendet, um die Suchergebnisse nach bestimmte Kriterien einzuschränken. Der meistverbreitete Filter ist Seiten-*Pagination*. Die andere Einsatzmöglichkeiten sind Zeit- oder Dokumententyp-Filter.

2.4.6 Gewichtete Suche

Im Kapitel *„Indexierung von Textdokumenten"* wurde bereits erwähnt, dass jedes Term von IR-System durch Indexierung gewichtet wird. Allerdings nicht nur Terme sondern auch die Suchanfragen(eng. search query) werden von IR-System gewichtet. Wenn die Suchanfrage aus mehrere einzelnen Suchanfragen besteht, dann wird die Gewichtung für alle einzelnen Anfragen ebenfalls berechnet.

Während die Volltextsuche die bessere Ergebnismenge durch Manipulation von Termen erreichen will, mischt sich die gewichtete Suche direkt in der Berechnung der Gewichtung ein. Dieses Verfahren ist als *Boosting* bekannt und wird meistens zur Anzeige von beliebtesten oder populärsten Dokumenten verwendet.

Es gibt zwei unterschiedliche *Boosting*-Verfahren: ***Field-Boosting*** und ***Term-Boosting***. ***Field-Boosting*** wird eingesetzt, wenn die Suche mehrere Felder anfragt und ein bestimmtes Feld höher als anderen priorisiert werden soll. Beim ***Term-Boosting*** wird die Suchanfrage durch Vorkommen eines bestimmten Terms höher gewichtet.

2.4.7 Autovervollständigung

Autovervollständigung ist heutzutage fast in jeder Anwendung vorhanden. Die Möglichkeit, durch die Eingabe von wenigen Zeichen auf vermutlich relevanter Suchbegriff zu kommen, hat sich in der Realität als sehr praktisch erwiesen. Grundsätzlich basiert dieses Verfahren auf die Ergänzung von eingegebenen Anfangsbuchstaben. Trotzdem gib es immer mehr Anwendungen, wo die eingegebene Buchstaben nicht als Anfangsbuchstaben sondern als einen Suchbegriff interpretiert werden und vermutlich relevante Dokumente sofort zum Auswahl angezeigt werden.

Mit Hilfe von Autovervollständigung wird das Erinnerungsproblem zu einem Wiedererkennungsproblem. Denn es reicht wenige Informationen, um ein gewünschter Suchbegriff zu treffen. Außerdem können auch Tippfehler bei der Autovervollständigung berücksichtigt und vermieden werden.

2.5 ElasticSearch

ElasticSearch ist ein Open-Source Suchserver, welcher auf Apache Lucene Bibliothek basiert. Das Projekt wird in der Programmiersprache Java geschrieben und ermöglicht die Kommunikation über REST-Schnittstelle. Die Daten werden in JSON-Format übertragen und in einem invertierten Index abgespeichert. Ein Index entspricht einem Datenbank im Datenbankmanagement System. Jeder Datensatz

wird durch Dokumenten und Spalten durch Felder repräsentiert. ElasticSearch verfolgt eine dokumentenorientierte Architektur und ist deswegen in der Lage auch komplexe Objekte in einem Feld zu speichern.

Relationale Datenbank	**Elasticsearch**
Datenbank	Index
Tabelle	Dokumenttype
Datensatz	Dokument
Spalte	Feld

Tabelle 2.1: Begriffsvergleich zwischen Relationale Datenbanken und Elasticsearch

ElasticSearch unterstützt alle im Kapitel 2.4 vorgestellten Methoden der Informationsbeschaffung und erweitert sie mit neuen Funktionalität. Zum Beispiel ist es möglich die Dokumente in Echtzeit zu indexieren. Die Indexierung wird dabei im Arbeitsspeicher gehalten und erst später auf die Festplatte geschrieben. Besonders mächtig ist ElasticSearch bei der Berechnung von Gewichtung(eng. score). Denn es werden mehrere Funktionen zur Manipulation von Score bereitgestellt. Es ist sogar erlaubt eigene Funktion zur Berechnung von Score zu schreiben. Standardmäßig wird aber Score nach Lucene-Formel berechnet[16].

$$score(q,d) = queryNorm(q) * coord(q,d) * \sum_{t(in)q} (tf * idf^2 * boost(t) * norm(t,d))$$

- $score(q,d)$ - Relevanz eines Dokumentes (d) für die Suchanfrage (q)
- tf - term frequency (Kapitel 2.4.2) - $tf = \sqrt{termFreq}$
- idf - inverse document frequency (Kapitel 2.4.2) - $idf = 1 + log(\frac{numDocs}{docFreq+1})$
- $queryNorm(q)$ - Funktion zur Berechnung der Normalisierung von *Search-Query* - $queryNorm(q) = 1/\sqrt{sumOfSquaredWeights}$, $sumOfSquaredWeights$ summiert IDF von allen Termen in der Suchanfrage
- $coord(q,d)$ - wird verwendet, um Dokumente mit mehrere vorhandenen Suchtermen höher zu gewichten $coord(q,d) = \frac{t(in)q}{t(in)d}$
- $boost(t)$ - erlaubt *boosting* von *Search-Query* (Kapitel 2.4.6). Standardmäßig ist die Wert gleich 1.
- $norm(t,d)$ - Funktion zur Berechnung von Feldnormalisierung $norm(t,d) = 1/\sqrt{numTerms}$, $numTerms$ steht für die Anzahl von Termen im gesuchten Feld

Die wichtige Funktion, die in diesem Zusammenhang auch erwähnt werden soll, ist die *Decay*-Funktion. *Decay*-Funktion berechnet die Relevanz eines Dokumentes Abhängig von dem Distanz zwischen einem nummerischen Feldwert und dem

[16] **www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html**.

benutzerdefinierten Wert. Das nummerische Feld kann eine Zahl, ein Datum oder Geo-Location sein. Die Distanz wird *linear*, *exponential* oder standardmäßig nach *Gaus*-Algorithmus berechnet.

$$S(doc) = exp(-\frac{max(0, |fieldValue - origin| - offset)^2}{2\sigma^2})$$

- $fieldValue$ - Feldwert
- $origin$ - Startwert, der bei der Distanzberechnung berücksichtigt wird. Beim Datum kann es zum Beispiel das heutiges Datum sein.
- $offset$ - Decau-Funktion wird berechnet, wenn die Distanz größer als $offset$. Standartwert ist 0.
- σ^2 -sorgt dafür, dass die berechnete Distanz in benutzerdefinierten Intervall liegt. $\sigma^2 = -\frac{scale^2}{2*log(decay)}$, $decay$ definiert den Abstand zwischen den Dokumenten. Standartwert ist 0.5. $scale$ ist die benutzerdefinierte Distanz.

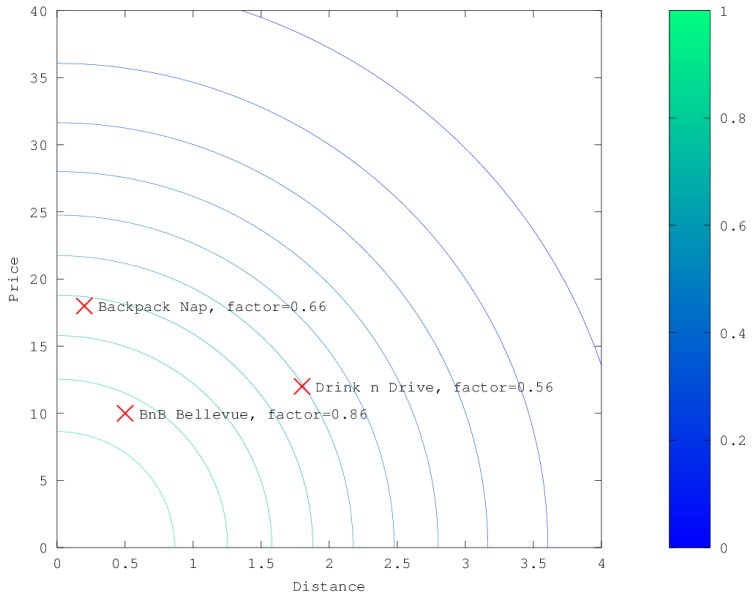

Abbildung 2.10: Visualisierung von *Decay*-Funktion mit *Gaus*

2.5.1 Sharding

Das Wort *Elastic* in dem Projektname bezieht sich auf horizontale Skalierbarkeit, welche ermöglicht, die Daten auf mehrere Server zu verteilen. Alle Daten werden in einem Cluster vorgehalten. Ein Cluster besteht aus einem oder mehreren Knoten (eng. Node), die auf unterschiedlichen Server laufen können. Die Synchronisierung zwischen den Knoten wird über Cluster gesteuert und kann relativ einfach und flexibel von Benutzer konfiguriert werden. Wenn ein Server besonders gut aufgerüstet ist, ist es sinnvoll ein *Master* und *Slave* Knoten festzulegen.

Um Ausfallsicherheit zu gewährleisten, werden die Dokumente der einzelnen Indizes in mehrere *Shards* repliziert und auf den Knoten verteilt. Der Shard, welcher das Dokument zuerst speichert, wird *primary shard* genannt. Alle weitere *shards* sind *replica shards* und behalten nur die Kopie von diesem Dokument. Wenn ein Node ausfällt, kann das Dokument aus dem *replica shard* eines anderen Nodes gelesen werden[Rafal Kuc 2013]. Die Abbildung 2.10 stellt ein Cluster mit zwei Knoten, 4 *primary shard* und 1. *replica shard* dar.

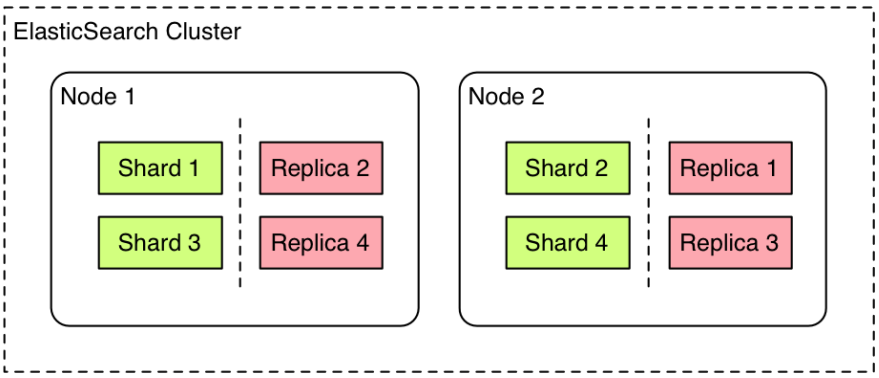

Abbildung 2.11: Beispiel - Ein Cluster mit zwei Knoten, 4 *shards* und 1 *replica shard*[Terrier 2013]

3. Systementwurf

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.0.2 Prozessablauf

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.0.3 Domain-Schicht

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine

falsche Anmutung vermitteln.

3.0.4 DAO-Schicht

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.0.5 Controller-Schicht

Unbedingt beschreiben, dass das Controller auf civis-tools zugreift, um die Daten aus dem Text zu extrahieren.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.0.6 Service-Schicht

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.0.7 Darstellung-Schicht

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4. Entwicklung und Implementierung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.1 Schnittstelle

?

4.2 Suchfunktionen

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.2.1 API

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest

gefburn"? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum" dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.2.2 Benutzeroberfläche

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext" oder „Huardest gefburn"? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum" dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

4.2.3 Softwarequalität

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext" oder „Huardest gefburn"? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum" dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

5. Evaluierung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

6. Problemen und Schwierigkeiten

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

7. Zusammenfassung und Ausblick

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Literaturverzeichnis

Dorschel, Joachim (2015). „Praxishandbuch Big Data - Wirtschaft – Recht – Technik". 1. Aufl. Berlin Heidelberg New York: Springer-Verlag (siehe S. 4, 6).

Jan, Steemann (2013). *Datenmodellierung in nicht relationalen Datenbanken*. url: https://entwickler.de/online/datenbanken/datenmodellierung-in-nicht-relationalen-datenbanken-137872.html (siehe S. 6).

OnPage (2016). *Tag Cloud*. url: https://de.onpage.org/wiki/Tag_Cloud (siehe S. 12).

Henrich, Andreas (2008). „Information Retrieval Grundlagen, Modelle und Anwendungen". 1. Aufl. Otto-Friedrich-Universität Bamberg (siehe S. 12 f.).

Mandl, Thomas (2001). „Tolerantes Information Retrieval". Hochschulverband für Informatinswissenschaft(HI)e.V (siehe S. 14).

Salton, G.; Wong, A.; Yang, C. S. (1975). „A Vector Space Model for Automatic Indexing". *Commun. ACM* 18.11, S. 613–620. doi: 10.1145/361219.361220. url: http://doi.acm.org/10.1145/361219.361220 (siehe S. 15).

Rafal Kuc, Marek Rogozinski (2013). „Mastering ElasticSearch". 1. Aufl. Packt Publishing Ltd. (siehe S. 20).

Terrier, Francois (2013). *On ElasticSearch performance*. url: https://blog.liip.ch/archive/2013/07/19/on-elasticsearch-performance.html (siehe S. 20).

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Chunker - Machine Learning Technologie - Teilt Text in syntaktisch korrelierten Teile von Wörtern, wie Nomen Gruppen.

Coreference Resolution - Machine Learning Technologie - Bezugnahme auf dieselbe Entität.

DAO - Data Access Object - Adapter zur Abstrahierung und Entkopplung von Datenzugriffe.

Invertierter Index - wird in Suchserver eingesetzt und bezieht sich auf die Tatsache, dass nicht die Dokumente auf Worte zeigen, sondern eine Liste von Worten auf Dokumente zeigt.

IR - Information Retrieval

Named Entity Recognition - Machine Learning Technologie - erkennt und klassifiziert Bestandteile im Text.

NoSQL - not only SQL - bezeichnet Datenbanken, die einen nicht-relationalen Ansatz verfolgen.

Part-Of-Speech tagging - Machine Learning Technologie - Das Zuweisen von Markierungen zu einzelnen Einheiten (Wortart-Annotierung).

SentenceDetector - Machine Learning Technologie - teilt Text in einzelne Sätze

Tokenizer - Machine Learning Technologie - teilt Sätze in einzelne Worte

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Datum: 3. Februar 2016, Berlin **Unterschrift:**